Bare Demo of IEEEtran.cls for IEEE Conferences

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—This work is focused on the prediction of affective states using non-obtrusive sensors. Non-obtrusive sensors are preferred in this work because of a hypothesis that states that an obtrusive sensor would alter the affective states of a user. Specifically, the method uses data gathered from a keyboard and mouse dynamics plugin embedded in a tutoring system designed for the teaching of computer programming in the Python language. The tutoring system requires from the user to practice their programming skills in order to progress through the course, and the users creativity plays an important role in the solving of programming exercises. Also a pre-processing method is applied, which generates feature vectors consisting of 39 features representing each interaction of the users by recording their keyboard and mouse dynamics. 149 feature vectors were generated and used as input of four different classification algorithms: Nave Bayes, J-48, k-NN, and Artificial Neural Networks. These classification models are then compared by their performances when classifying learners into five different affective states: boredom, frustration, distraction, relaxation and engagement. Each of the classification models achieved an accuracy of around 75% and kappa of around 0.45, and, in average, J-48 was the better algorithm with an accuracy of 75.97%. Results show that the data gathered from the non-obtrusive sensors can successfully be used as another input to classification models in order to predict an individual's affective states during their interaction with a programming tutoring system.**

I. Introduction

Introductory programming courses are generally regarded as difficult [1], [2] and there is a common conception that they often have a high failure rate [3]. There are multiple factors involved. Jenkins [4] argues that some are deeply related with the expectations, attitudes, and previous experiences of the teaching staff and students. Another factor is the nature of the subject that involves the learning of new abstract constructs, syntax and tools. Also groups are often heterogeneous and thus it is difficult to design courses that are beneficial for everyone. Many students begin to program when they are at their first year of university and until then are confronted with a totally new topic that does not respond to their habitual study approaches. Dijkstra [5] argues that the subject of programming is very problem-solving intensive, and it requires high precision because even the slightest perturbation can render a program totally worthless. These difficulties often frustrate students. Jenkins [4], [6] notes that many students expect the course to be difficult and come with the notion that they will have to struggle, others may have a stereotyped image of a programmer, these beliefs can negatively affect their initial motivation. During the course, novice programmers experience many emotions, for instance frustration and confusion when they cannot find a bug in a program, but also joy when they successfully run a challenging program for the first time. They can also become bored if they found the exercises too repetitive or too easy. Learning to program is a difficult task, where emotions play a significant role. Research has gone far to understand the role of emotion in learning, both in the process and the outcome; for instance, in e-learning [7], [8] and in programming [9], [4], [10], [11].

It is expected that flow/engagement is the ideal affective state in which students tend to be most capable of acquiring meaningful information through the learning process. Flow is defined by Csikszentmihalyi [12] as a mental state in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment. Engagement is also a positive affect where students are thought to be more involved behaviourally, intellectually, and emotionally in their learning tasks [13]. When designing instruction, an objective is to assign learning activities that result challenging to students, but not much as to frustrate them. For this, experienced instructors gauge a students affective state and then assign an activity with the appropriate level of difficulty. In affect-aware intelligent learning environments (ILEs), emotions have a central role in the students' user model. However, recognition of emotions is a difficult task, even for humans. Instructors use their social intelligence often to recognize students affective states. In class a good instructor habitually reads the faces of students in the classroom to see if they are confused, bored or engaged; then decides what to do next. ILEs can be enhanced if they can offer an automatic perception of emotions. The recognition and simulation of human affects are becoming important fields of study, as many researchers have demonstrated affect-aware computers can provide better performance in assisting humans [14]. When ILEs embrace these methodologies and techniques a problem arises. In order to perform recognition of the users' affective states, sensors must be used to gather data. Some sensors rely on physiological readings and must

be in physical contact with students. These sensors can be considered intrusive or invasive, and can disrupt the students learning experience [15], [16], [17]. Other sensors such as video cameras can also be considered invasive, since they always transmit the identity, appearance and behaviour along with emotional data [14].

In this work, a method for the recognition of affective states through keystroke and mouse dynamics is proposed. The hypothesis is that students vary the dynamic of keystrokes according to their affective state when programming. The method has three main components: a Javascript library to capture keyboard and mouse dynamics from a browser-based editor, a pre-processing step whose output is a feature vector that is sent to the third component, a classification algorithm. An experiment was conducted where students solved a series of programming exercises using a web based learning environment. Using only the data from the students keyboard and mouse dynamics six affective states where recognized for each of the attempts at solving the exercises. Results obtained from the experiment are promising. The affective states recognized include the most common in novice programmers according to the study of Bosch, D'Mello & Mills [18]: boredom, engagement/flow, confusion and frustration. For this experiment four binary classifiers were compared: J-48 decision trees, k-nearest neighbour classifier, feed forward neural network, and a nave Bayes. The output of each classifier determined if the student experienced the affective state during the exercise. The proposed method could be used in an ensemble with other sensor channels to improve the non-invasive recognition of the students affective states when they are working on a programming task. In order to determine what affective states a student was experiencing, an Experience Sampling Method (ESM) was used by Csikszentmihalyi & Larson [19]. After the students successfully solve a programming exercise, they are presented with an ESM survey that asks what they were feeling during their solving of the exercise.

If a relationship could be modelled, ILEs could adapt the instruction using the same devices required to input the program to the computer. This could be important because writing programs in a computer is an important learning activity when learning programming. In study by Lahtinen, Ala-Mutka & Jrvinen [2] is reported that students rated "working alone on programming coursework" as a more useful situation than lectures.

The structure of this work is organized as follows: Section Related Work presents a series of works related to the proposed method in this paper; Section Proposed Method describes the proposed method for the recognition of affective states in a web based learning environment for the teaching of programming languages; Section Experiment explains the experimental evaluation of the proposed method, following by a Results sections and finally, a Conclusion and Future Work are discussed.

II. Related Work

Affect recognition is an active field of research. Many methods have been proposed, some require the intervention of the user to fill up questionnaires or forms; these selections remain static until the user changes the values. These methods are easy to implement, but cannot detect dynamic changes. A more dynamic approach requires the use of sensors to capture affective states as they change. The context, the environment and the learning activity determine what kind of sensors can be used. The most common learning environment is the classroom, a physical space with context to facilitate learning. But learning is possible in a wide variety of settings, such as outside-of-school locations and outdoor environments these are sometimes referred as ubiquitous learning environments, an example of such environments is the work of Yang [20] in a context-aware environment, but missing affective information. There are also virtual learning environments such as the one proposed by Dillenburg [21] where learners can have avatars, and virtual places where they can play roles and socialize.

In the general context of learning there have been some approaches for affect recognition. The work of Kapoor y Picard [22] uses a multi modal approach using sensory information from facial expressions and postural shifts of the learner combined with information about the learner's activity on the computer; the learning activity was solving a puzzle on a computer. They report using multimodal Gaussian Process approach achieving an accuracy of over 86%. Sidney, et al., [16], [**?**] enhances the intelligent tutor system AutoTutor with affective recognition using nonintrusive sensory information from facial expressions, gross body movements, and conversational cues from logs. The subject material of that system consisted in lectures of computer literacy.

Elliott, Rickel y Lester [23], [24] propose the integration of an affective reasoner with an agent in a virtual environment. In this work an agent called Steve responds emotionally to other agents and interactive users. The agent simulates his emotions through different multimedia modes including facial expressions and speech. Affective reasoning agents were used for training, by putting students in work related situations. Agents could react to the behavior of students, for instance if a student was being careless in a task and was in a dangerous situation, Steve would show distress or fear. Also in a virtual environment the work of McQuiggan, Robison & Lester [25] extends this line of research by investigating the affective transitions that occur throughout narrative-centered learning experiences. The analysis of affective state transitions in this work replicated the findings by DMello et al. [24] and Baker et al. [9] where also engagement/flow dominated self-reported affect. For outdoor environments Shen, Wang & Shen [26] augment a pervasive e-learning platform with affective recognition. The results about emotion recognition from physiological signals achieved a best-case accuracy (86.3%) for four types of learning emotions.

Bosch, D'Mello & Mills [10] analysed the relationship between affective states and performance of novice programmers

when they were learning the basics of computer programming in the Python language. The results of their study indicated that the more common emotions students experienced were engaged (23%), confusion (22%), frustration (14%), and boredom (12%). It was useful to consider these results, as it presented evidence of what affective states need to be targeted in order to obtain less biased data. For example, if a less common emotion was chosen, a classifier could opt to classify any feature vector as not experiencing such emotion. Nevertheless, the classifier would obtain accurate results, although the classifier would be inaccurate at determining if a feature vector was actually experiencing the given affective state. Similar to the previous work, Rodrigo et al., [9] observed which affective states and behaviours relate to student's achievement within a basic computer science course. The authors found that confusion, boredom and engagement in IDE-related (on-task) conversation are associated with lower achievement.

There has been very little research reported on the effectiveness of the use of keyboard and mouse dynamics as a sensory channel for affective recognition, and the few have not been focused on programming. The preliminary work by Zimmermann et al. [27] describes a method to correlate users interactions (keyboard and mouse) with an emotional state. Several physiological parameters were measured concurrent with the task. The parameters included respiration, pulse, skin conductance level, and corrugator activity. The task assigned to subjects was to shop on an e-commerce website for office-supplies and finally write a predefined message to the shop operator. Subjects where watching videos in order to change their emotional state during the experiment. The method was validated using self-reported emotions using the self-assessment-manikin (SAM), devised by Lang [28], designed to assess the dimensions valence, arousal and dominance directly by means of three sets of graphical manikins. Results show that the film clips were effective in inducing the expected mood changes, but no further empirical results are presented. The work of Vizer, Zhou & Sears [29] also uses sensory data based on the time elapsed between each key press, the task given to users was to write a free-text and used linguistic features in order to recognize both physical and emotional stress. The results show a classification accuracy of 62.5% for physical stress and 75% from emotional stress; authors argue that these results are comparable with other approaches in affective computing. They also stress that their methods must be validated further in other contexts. Moods may have an impact on programmers performance according to Khan, Hierons y Brinkman [11]. It may be possible to detect moods on the basis of information regarding the programmers use of the keyboard and mouse, and to integrate them into development environments that can improve programmer performance. They briefly describe a further experiment that could use keyboard and mouse but only as future work. There are other studies about the behaviour of programmers, which are not directly concerned with affective states but are nevertheless important to their performance. Eye tracking in computing education is proposed in the work Busjahn et al., [30], and it is also used for assessing learners comprehension of C++ and Python programs in Turner et al., [31]. Blikstein [32] proposed an automated technique to assess, analyse and visualize students learning computer programming. Blikstein employs different quantitative techniques to extract students behaviours and categorize them in terms of programming experience.

Research works related to Keystroke Dynamics (KD) are carried out either using fixed-texts, or free-texts Gunetti y Picardi [33]. KD performed on fixed-texts involves the recognition of typing patterns when typing a pre-established fixed-length text, e.g., a password. In the other case, free-text KD achieves the recognition of typing patterns when typing a text of arbitrary-length, e.g., a description of an item. However, as noted by Janakiraman y Sim [34], most of the research regarding KD is done on fixed-text input, the reason being that fixed-text KD usually yields better results than free-text KD. Yet, the authors of this work share the opinion with Janakiraman, R., and Sim, T., that it would be more useful if KD can handle free text as well as fixed text, this is also a requirement if the text is a program.

Although the use of Keystroke Dynamics (KD) is found in several research works as a biometric measure, its use for identifying affective states is rare in comparison. Epp, Lippold & Mandryk [35] effectively used KD in conjunction with decision-tree classifiers for the identification of 15 affective states. Although their work was based on fixed-text their technique to extract a feature vector was an inspiration for the proposed method in this work. As for free-text KD, Bixler y D'Mello [18] present a method for the identification of boredom and engagement based on several classification models.

Regarding Mouse Dynamics (MD), some research has been conducted for the identification of affective states, although, as with the case of KD, MD is mainly used as a biometric measure for authentication. Salmeron-Majadas, Santos and Boticario [36] use both MD and KD to predict four affective states using five different classification algorithms. Bakhtiyari y Husain [37] discuss a method based on fuzzy models for the recognition of emotions through KD, MD and touch-screen interactions. For a broad review of emotion recognition methods based on KD and MD, the work by Kolakowska [38] is recommended.

III. Proposed Method

The goal of this work is to propose an affective recognition method based on the sensory data provided by the keyboard and mouse dynamics generated by a learner as he/she types a programing exercise. The main components of the method are depicted in Figure 1. A brief explanation of the whole process is explained next, details come later. The process starts when the learner begins to type a program in a browser-based editor. As she types or moves the mouse the dynamics are recorded. When the learner submits the code to evaluation, the request includes the sensory data along with the code and information about the session. In the server, the code is evaluated in an

external virtual machine that provides a sand box to prevent malicious or erroneous code to halt the server. When the result is ready, it is recorded along with the sensory data and sent to the preprocessing module. The output is a feature vector, ready for classification. A previously trained classifier is responsible for the classification, which outputs the predicted affective state. The method does not consider other sensory data, but it could be integrated with other sensory inputs in a multi-modal approach. Each of the steps is explained in detail next.

1) Capturing the Keystroke and Mouse Data: While a student is trying to solve an exercise, a script coded in JavaScript is running in the background, which captures every keystroke, mouse movement and mouse button press. Each capture of these events records a timestamp in milliseconds (using the method getTime() of Javascript's built-in class Date) that describes when the event occurred. If the event is a keystroke, the script captures what key was specifically pressed, and what type of event occurred, it can be either a key- down or a key-up event. If it is an event related to a mouse button press, the key code of that button is recorded, as well as the type of event occurred again key-down or key-up. Finally, if the event was a mouse movement, the mouse coordinates inside of the web browser are recorded. The script monitors the mouse position every 100 milliseconds, and only if the position has changed, it records the new position. Each time a learner tries to evaluate a program, all the data generated is sent to the server along with the code. When the result of the execution is returned, all records are cleared and the process starts again. There is no problem if the user leaves for a long period of time, because no event will be triggered. If a user copies and then pastes the code from another source, this will be recorded. There is a limitation, only the browser-editor must be used; this could be a problem for more advanced programmers needing specialized editors. On the other hand a browser-based editor with the corresponding remote execution, does not require the installation of interpreters or compilers in learners computer. The code could even be written in a mobile device or any web-enabled device. The interface of the web-based editor is shown in Figure 2. Programming exercises are evaluated using unit tests; the results of the evaluation are shown to users. If all tests the program is considered to be correct. An example execution is shown in the left side image of Figure 2. In this example the learner was asked to write a function to add two numbers, in Python. The source code for the Protoboard web based learning environment including the KD and MD functions are open source and available as Github repositories at http://git.io/vJlUV also the code for the sandbox is in http://git.io/vJlUj.

2) Preprocessing of the Keystroke and Mouse Data: The raw data obtained from the script needs to be preprocessed to obtain a feature vector. Basically, this pre-processing consists in measuring the delays between key- down, key-up or mouse-move events triggered during an exercise. These events have the dynamic shown in Figure 3; for example when typing the word key a user first presses the letter K and triggers the key-down event, this is indicated with an arrow that changes the state of the key, the time of the event is important and also recorded. Only the event data is received from the browser, in order to generate feature vectors, the patterns and rhythm users have when pressing consecutive keys are captured measuring the delays between events, as it is a common practice when dealing with keystroke dynamics. In this work, the definitions proposed by Sim y Janakiraman (2007, June) are used. Held time (Ht) is defined as the time between a key-down and a key-up of the same key, this would be Ht(K) in the example. Inter key time (It) is defined as the time between two consecutive key-down events, this time could be negative is the second key is pressed before the first is released. A sequence is defined as a list of consecutive keystrokes. In the above example valid sequences could be [K, E], [E, Y] and [K, E, Y], the first two are digraphs and the third is a trigraph. While there can be sequences of any size in a text, working only with digraphs and trigraphs is preferred. Even if only digraphs and trigraphs are considered, the amount found in free-text is too large and not very useful for a feature vector as noted by Epp, Lippold y Mandryk (2007) so they propose to use statistical measures to capture the patterns; the same strategy is used in this work. The averages and standard deviations are calculated from the delays between the events of the digraphs and trigraphs in each program.

To calculate the average and standard deviations of these presses, the delays between a key-down and a key-up event of the left button clicks are used. In addition to these averages and standard deviations of the delays between keystrokes and mouse button presses, the average and standard deviations of the number of total events contained in a digraph and a trigraph are calculated. These features are proposed and explained by Epp, Lippold y Mandryk [35]. Most of the times, a digraph should contain four events, while a trigraph six. However, sometimes an individual can start a digraph or a trigraph before ending the previous one. These additional features represent these particular cases, and could be meaningful for the estimation of a learner's affective states. Regarding the mouse movements, the average and standard deviation of the duration of each mouse movement, and the averages and standard deviations of the movements in the axes X and Y are also calculated. Lastly, a final feature is added to preprocessing of the data. The web tutorial recorded how many attempts a student required before successfully solving an exercise. This number of attempts is also included in the feature vector. The final feature vector consists of 39 features; these are based on the work of Epp, Lippold & Mandryk [35] and are shown in Table 1. (The code used for this step is available at: https://github.com/amherag/keyboard-mouse-dynamics).

Once feature vectors are obtained from an experiment, the generated dataset is normally used as training data for a classifier. Researchers of affective recognition have used a wide variety of classification algorithms. As a proof of concept four well known classification algorithms where compared: k-Nearest Neighbors (k-NN) algorithm, a feed forward neural network trained with back- propagation, a nave Bayes classifier and finally a decision trees algorithm for rational data (J-

48). Details for these algorithms, can be found in the textbook by Tan, Steinbach, & Kumar [39]. The accuracies obtained by these algorithms as well as the parameters used are reported in the next section.

IV. Experiment

The aim of this research is to evaluate the use of keyboard and mouse dynamics as an appropriate sensory input for an affective recognition system. The context of use is a learning environment for programmers. In particular for learning activities consisting in writing short programs interactively. An experimental approach was adopted with this aim. Sensory and quantitative data was collected from learners as they were enrolled in a basic course of Python programming. This data was then pre-processed using the method described earlier, and together with the quantitative data obtained from users, classifiers were trained and validated. The results were then compared and came to conclusions regarding the effectiveness of the method. The goal given to subjects was to solve as many exercises as they could in a period of two weeks. There was neither time limit nor a minimum amount of time required for a participant while trying to solve the exercises or complete the tutorial. The participants were able to stop and resume their interaction with the system at any time.

Figure 4. Web-based editor used for code evaluation. A tutorial was developed to obtain the necessary data using Protoboard, a web-based programming learning environment (the latest version can be found online at http://python.protoboard.org/). Protoboard has the functionality described in the proposed method and shown schematically in Figure 1. Users-log in to Protoboard using their account credentials from a popular social network (Facebook). The web tutorial begins with three introductory videos that explain the fundamentals of programming in Python, and how to solve the programming exercises in the course. What follows after these videos, are 13 programming exercises that the students need to solve in consecutively. Exercises are of incremental difficulty. In Figures 2 and 3, the interface of this platform is presented. For this experiment Protoboard was configured to allow unlimited tries to solve an exercise.

A total of 55 volunteers, with no previous experience in Python, where recruited as a response to three announcements in a special interest group of programming students in a social network. Learners were required to login with their Facebook credentials, this had the advantage of not requiring the system to maintain users credentials or profile information as this is kept outside in an external service. Unfortunately this decision also kept some prospects from volunteering; a subject reported that he normally creates temporary email accounts to try new web services. Other requirement was that they where all adults, this age is 18 years in Mexico where the experiment was conducted. The requirement of the social network account stopped several prospects from volunteering, but on the other hand gave researchers access to public information about the subjects. All the learners were either current students or graduates of Computer Systems Engineering. Their ages were in the range of 18 to 30 years old. Although no experience in software programming was needed, as the web tutorial's course is of a very basic level, all the participants were required had completed at least one course in software programming. Results from the initial survey are shown in Figure 4.

In order to determine what affective states a student was experiencing, an Experience Sampling Method (ESM) was used (Csikszentmihalyi y Larson, 1987). After the students successfully solve a programming exercise, they are presented with an ESM survey that asks what they were feeling during their solving of the exercise. A very brief description is given about what to do in this survey, followed by statements the students need to answer according to how they were feeling. As an example, the statement I was feeling frustrated is presented, and a student needs to answer either Strongly agree, Agree, Neutral, Disagree, and Strongly Disagree.

After the two weeks of the experiment, only four learners completed all of the programming exercises and 22 did not completed any. Out of the total activities available (videos, survey and questionnaires) only two completed all. Figure 5 shows the number of exercises and activities completed by each learner.

Figure 5. Web-based editor used for code evaluation.

Figure 6. Class distribution of emotions as reported by learners. The participants' interaction generated a total of 142 feature vectors one for each successfully completed exercise. The affective states reported by learners after completing each exercise was grouped in three classes: Yes, Neutral and No. The class distribution of the emotions reported is shown in Figure 6. This results show that the most common emotions were flow/engagement (72%) and relaxation (61%) while few learners reported distraction (5%), frustration (8%) and boredom (2%). This distribution is different from what was reported by DMello et al. (2013) and Baker et al. (2007). Although Flow/Engagement was also the predominant class, the distribution is skewed to the first two. There are some possible reasons for this; first the majority of students had experience in programming, so learning new one is not that problematic, a second reason could be the freedom users had to abandon the activities, perhaps frustrated or bored learners simply quit the tutorial. A high number of learners (49%) did not completed any exercise and the once who completed more where also the more experienced.

Classifiers where implemented using the software Rapid Miner using a feature selection module as part of the process. A genetic algorithm was used in the user selection module. In the end, the subset for the frustration classifier consists of 11 features, and the subset for the boredom classifier consists of 13 features. The performance of the classifiers is shown in Table 1. Table 1. The components of the feature vector; adapted from Epp, Lippold y Mandryk (2007). Nave Bayes J-48 k-NN ANN Affect Accuracy k Accuracy k Accuracy k Accuracy k Flow/Engaged 79.00% (0.11) 0.50 80.48% (0.10) 0.51 76.14(0.09) 0.43 78.43(0.12) 0.45 Relaxation 71.10% (0.09) 0.40 72.57% (0.11) 0.45 69.14%(0.08) 0.37

Fig. 1. Simulation results for the network.

71.24%(0.07) 0.34 Distraction 70.33% (0.13) 0.43 71.00% (0.10) 0.43 74.00%(0.06) 0.50 72.52%(0.07) 0.39 Frustration 74.71% (0.09) 0.49 73.86% (0.12) 0.45 72.62%(0.09) 0.42 78.0%(0.09) 0.51 Boredom 74.71% (0.13) 0.47 84.57% (0.06) 0.59 83.81%(0.06) 0.58 76.19%(0.09) 0.45

The results obtained with this method are satisfactory, considering a free-text was used. The accuracies and kappa coefficients obtained are close to what is usually obtained in fixed-text methods, for example, the results presented in Epp, Lippold y Mandryk (2007). In the case of the kappa coefficients, it is usual to see values below 0.2 in methods involving free-text. In this case, some values of kappa were close to 0.5, is important to consider the values of kappa in these results because the class distribution is not uniformly distributed. As it is observed in many works in affective recognition, decision trees normally produce competitive accuracy. In this case the J-48 classifier obtained an accuracy of 80.48% marginally better than the others, and an accepted kappa statistic for the kind of problem. Artificial neural network give the highest accuracy in the classification of frustration.

V. Conclusion

Acknowledgment

This received funding from Tecnlogico Nacional de Mxico through the project: Tcnicas de computacin inteligente para el secuenciado adaptativo de ejercicios de programacin en la nube.

References

[1] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer science education*, vol. 13, no. 2, pp. 137–172, 2003.

[2] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," in *ACM SIGCSE Bulletin*, vol. 37, no. 3. ACM, 2005, pp. 14–18.

[3] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *ACM SIGCSE Bulletin*, vol. 39, no. 2, pp. 32–36, 2007.

[4] T. Jenkins, "The motivation of students of programming," in *ACM SIGCSE Bulletin*, vol. 33, no. 3. ACM, 2001, pp. 53–56.

[5] E. W. Dijkstra *et al.*, "On the cruelty of really teaching computing science," *Communications of the ACM*, vol. 32, no. 12, pp. 1398–1404, 1989.

[6] T. Jenkins, "On the difficulty of learning to program," in *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, vol. 4. Citeseer, 2002, pp. 53–58.

[7] B. Kort, R. Reilly, and R. W. Picard, "An affective model of interplay between emotions and learning: Reengineering educational pedagogy-building a learning companion." in *icalt*, vol. 1, 2001, pp. 43–47.

[8] D. Rossin, Y. K. Ro, B. D. Klein, and Y. M. Guo, "The effects of flow on learning outcomes in an online information management course," *Journal of Information Systems Education*, vol. 20, no. 1, p. 87, 2009.

[9] M. M. T. Rodrigo, R. S. Baker, M. C. Jadud, A. C. M. Amarra, T. Dy, M. B. V. Espejo-Lahoz, S. A. L. Lim, S. A. Pascua, J. O. Sugay, and E. S. Tabanao, "Affective and behavioral predictors of novice programmer achievement," in *ACM SIGCSE Bulletin*, vol. 41, no. 3. ACM, 2009, pp. 156–160.

[10] N. Bosch, S. DMello, and C. Mills, "What emotions do novices experience during their first computer programming learning session?" in *International Conference on Artificial Intelligence in Education*. Springer, 2013, pp. 11–20.

[11] I. A. Khan, R. M. Hierons, and W. P. Brinkman, "Mood independent programming," in *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!* ACM, 2007, pp. 269–272.

[12] M. Csikszentmihalyi, "Flow: The psychology of optimal performance," *NY: Cambridge UniversityPress*, 1990.

[13] R. L. Bangert-Drowns and C. Pyke, "Teacher ratings of student engagement with educational software: An exploratory study," *Educational Technology Research and Development*, vol. 50, no. 2, pp. 23–37, 2002.

[14] R. W. Picard, E. Vyzas, and J. Healey, "Toward machine emotional intelligence: Analysis of affective physiological state," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 10, pp. 1175–1191, 2001.

[15] J. Zhai and A. Barreto, "Stress detection in computer users through non-invasive monitoring of physiological signals," *Blood*, vol. 5, no. 0, 2008.

[16] K. D. Sidney, S. D. Craig, B. Gholson, S. Franklin, R. Picard, and A. C. Graesser, "Integrating affect sensors in an intelligent tutoring system," in *Affective Interactions: The Computer in the Affective Loop Workshop at*, 2005, pp. 7–13.

[17] I. Arroyo, D. G. Cooper, W. Burleson, B. P. Woolf, K. Muldner, and R. Christopherson, "Emotion sensors go to school." in *AIED*, vol. 200, 2009, pp. 17–24.

[18] R. Bixler and S. D'Mello, "Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits," in *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 2013, pp. 225–234.

[19] R. Kubey, R. Larson, and M. Csikszentmihalyi, "Experience sampling method applications to communication research questions," *Journal of communication*, vol. 46, no. 2, pp. 99–120, 1996.

[20] S. J. Yang, "Context aware ubiquitous learning environments for peer-to-peer collaborative learning," *Educational Technology & Society*, vol. 9, no. 1, pp. 188–201, 2006.

[21] P. Dillenbourg, D. Schneider, and P. Synteta, "Virtual learning environments," in *3rd Hellenic Conference" Information & Communication Technologies in Education"*. Kastaniotis Editions, Greece, 2002, pp. 3–18.

[22] A. Kapoor and R. W. Picard, "Multimodal affect recognition in learning environments," in *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 2005, pp. 677–682.

[23] C. Elliott, J. Rickel, and J. Lester, "Lifelike pedagogical agents and affective computing: An exploratory synthesis," in *Artificial intelligence today*. Springer, 1999, pp. 195–211.

[24] S. DMello, T. Jackson, S. Craig, B. Morgan, P. Chipman, H. White, N. Person, B. Kort, R. el Kaliouby, R. Picard *et al.*, "Autotutor detects and responds to learners affective and cognitive states," in *Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems*, 2008, pp. 306–308.

[25] S. W. McQuiggan, J. L. Robison, and J. C. Lester, "Affective transitions in narrative-centered learning environments." *Educational Technology & Society*, vol. 13, no. 1, pp. 40–53, 2010.

[26] L. Shen, M. Wang, and R. Shen, "Affective e-learning: Using" emotional" data to improve learning in pervasive learning environment." *Educational Technology & Society*, vol. 12, no. 2, pp. 176–189, 2009.

[27] P. Zimmermann, S. Guttormsen, B. Danuser, and P. Gomez, "Affective computinga rationale for measuring mood with mouse and keyboard," *International journal of occupational safety and ergonomics*, vol. 9, no. 4, pp. 539–551, 2003.

[28] P. J. Lang, "Behavioral treatment and bio-behavioral assessment: Computer applications," 1980.

[29] L. M. Vizer, L. Zhou, and A. Sears, "Automated stress detection using keystroke and linguistic features: An exploratory study," *International Journal of Human-Computer Studies*, vol. 67, no. 10, pp. 870–886, 2009.

[30] T. Busjahn, C. Schulte, B. Sharif, A. Begel, M. Hansen, R. Bednarik, P. Orlov, P. Ihantola, G. Shchekotova, M. Antropova *et al.*, "Eye tracking in computing education," in *Proceedings of the tenth annual conference on International computing education research*. ACM, 2014, pp. 3–10.

[31] R. Turner, M. Falcone, B. Sharif, and A. Lazar, "An eye-tracking study assessing the comprehension of c++ and python source code," in *Proceedings of the Symposium on Eye Tracking Research and Applications*. ACM, 2014, pp. 231–234.

[32] P. Blikstein, "Using learning analytics to assess students' behavior in open-ended programming tasks," in *Proceedings of the 1st international conference on learning analytics and knowledge*. ACM, 2011, pp. 110–116.

[33] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 3, pp. 312–347, 2005.
[34] R. Janakiraman and T. Sim, "Keystroke dynamics in a general setting," in *International Conference on Biometrics*. Springer, 2007, pp. 584–593.
[35] C. Epp, M. Lippold, and R. L. Mandryk, "Identifying emotional states using keystroke dynamics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 715–724.
[36] S. Salmeron-Majadas, O. C. Santos, and J. G. Boticario, "Exploring indicators from keyboard and mouse interactions to predict the user affective state," in *Educational Data Mining 2014*, 2014.
[37] K. Bakhtiyari and H. Husain, "Fuzzy model on human emotions recognition," *arXiv preprint arXiv:1407.1474*, 2014.
[38] A. Kołakowska, "A review of emotion recognition methods based on keystroke dynamics and mouse movements," in *2013 6th International Conference on Human System Interactions (HSI)*. IEEE, 2013, pp. 548–555.
[39] P.-N. Tan *et al.*, *Introduction to data mining*. Pearson Education India, 2006.